Reflowable Document Images for the Web

Thomas M. Breuel[1]
PARC, Inc., Palo Alto, CA, USA

Abstract *The paper describes on-going work on a system that transforms page-oriented document images into "reflowable document images", representations of the page image in HTML format that allows it to adapt to display devices of different sizes while preserving the original appearance of the image as much as possible and avoiding OCR errors. The approach to document layout analysis used by the system is outlined and the strengths and limitations of HTML for this application are discussed.*

1 Introduction

Large numbers of documents are formatted for printing on letter size or A4 paper. Those documents may be in PostScript, PDF, or TIFF formats. Sometimes they are generated electronically, at other times, they are scanned.

A number of techniques for putting these documents on the Web have been explored. Many of them are made available simply in their page oriented formats, as TIFF, PDF, or PostScript files. When displayed on a normal desktop screen, however, fonts usually become much too small when full pages are displayed, and users end up having to scroll up and down to read multi-column documents. TIFF, PDF, and PostScript are also not native web formats and require plug-ins or helper applications to view, which take some time to start up or may not be installed at all. Additionally, PostScript files from untrusted sources present a potential security risk. Many users prefer to simply print these documents for reading and then throw away the printed copies afterwards.

When the source of the document is available (e.g., a Microsoft Word or LaTeX file), there are special converters to turn these documents into HTML. For example, the popular LaTeX2HTML converter generates a web of linked HTML pages representing the source document. However, given the limitations of HTML, the resulting conversions are not entirely satisfying. First of all, LaTeX2HTML cannot handle many kinds of macro definitions and other features. In addition, most current web browsers cannot render mathematics; therefore, LaTeX2HTML generates a mixture of running text with bit-mapped images interspersed, which represent the mathematical formulas occurring in the source text. This may sometimes work reasonably well, but since web browser make no guarantees about the fonts or font sizes used for rendering text, the font sizes and appearances of the text may not match the size and appearance of the mathematical formulas at all.

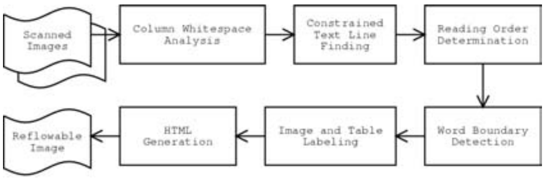


Figure 1. The layout analysis used in the generation of reflowable images. See the text for details.

The Google web site (`http://www.google.com/`) uses an interesting technique for rendering PDF documents: the characters in the PDF document are identified, and they are then placed at absolute locations within an HTML page, using the style sheet features of HTML. This results in a document that can be rendered in modern browsers and usually allows the reader to read most of the page content, but it may result in text lines overlapping and missing mathematical symbols. The appearance of the original text is not preserved, and the rendition remains page oriented and of fixed size.

None of these systems guarantee for a convenient and natural reading experience. What we would like to have is a way of turning arbitrary page-oriented documents back into a structured form that permits reflowing (reflowing is the adaptation of text to different window or page sizes).

One approach to this is to use an OCR (optical character recognition) system. Commercial OCR systems actually perform two major tasks. First, they recognize each individual character on the page, together with its font and other attributes. Second, they analyze the logical structure of the page (document layout analysis). They then combine the results from those two analyses to attempt to obtain a structured source document (e.g., in HTML, SGML, LaTeX, Microsoft Word, or other formats) that, when processed, will result in the same page-oriented document that the system was originally given.

[1] This paper describes joint work with Bill Janssen, Kris Popat, and Henry Baird.

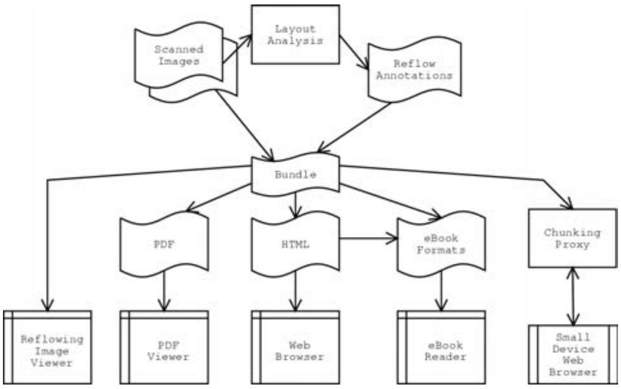


Figure 2. Different encodings that can be derived from the scanned images and reflow annotations.

Unfortunately, OCR systems are far from being able to achieve that goal in general. While they perform quite well on purely textual documents with simple layouts, they fail when pages contain many fonts, foreign languages, complex layouts, mathematics, chemical formulas, or many kinds of diagrams. In fact, the structured document format into which they output the result of the OCR analysis may not even be capable of representing all the features of the source document.

In our work on reflowable document images, we are taking an approach that falls in between attempting full OCR and just displaying the original page-oriented rendition. Our system performs a limited form of document layout analysis to decompose the document image into components that can then be reflowed. Usually, these components are word images and illustrations. This decomposition can then be transformed into a number of formats (shown in Figure 2) and viewed in a variety of different ways. Of particular interest are renditions in standard HTML format, because they can be viewed on almost any web browser.

Some aspects of the system, and a somewhat different approach to the layout analysis problem, have been previously described in [2, 3]. This paper describes on-going work and covers in more detail some of the experiences we have had with the system, as well as problems specifically with representing reflowable images using HTML.

The rest of the paper will briefly outline the current layout analysis component of the system, and describe direct viewing of reflowable image content. Then, it will present a more detailed discussion about the use (and limitations) of HTML as a format for displaying reflowable image content.

2 Layout Analysis

The details of the layout analysis method used in the system are described in [1]. Let us briefly outline how it works here.

Layout analysis begins with finding whitespace column boundaries; [1] describes a method for identifying these using a simple branch-and-bound algorithm and an evaluation function that is more robust than those used in previous white-space analysis methods.

Once column boundaries (if any) have been identified, the system identifies text lines. For that, a globally optimal constrained text line finder is used; the algorithm finds text lines that do not cross any of the identified column boundaries.

The output of these first two steps is a collection of text line segments. Each text line segment represents a continuous sequence of characters from the source documents in reading order. However, the reading order among the text line segments still needs to be determined. For that, the system takes advantage of a partial order among text line segments that can be determined easily and quite reliably based on certain geometric arrangements of text line segments. This partial order is then extended into a total order using a topological sorting algorithm.

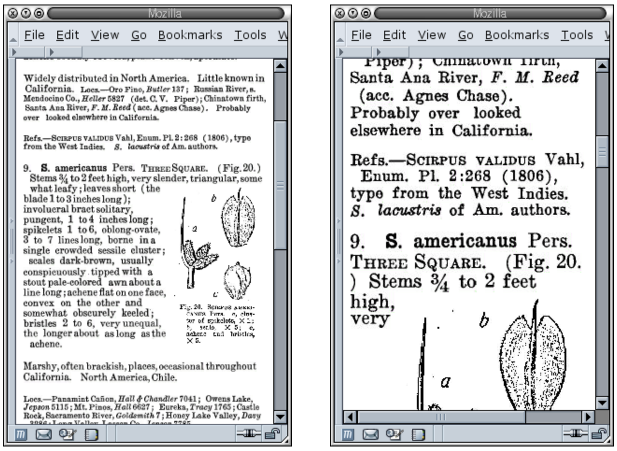
The output of layout analysis is then a set of text line segments in reading order. When these are rendered in reading order, one line at a time, they can be read and understood. (In this process, certain "floats" components of the original page image, such as figures and their captions, are put into a reading order as well, although, by the nature of such floats, they can be moved to some other locations without affecting readability.)

Prior to any layout analysis, parts of the page that likely represent tables, figures, or images need to be identified. The current system relies on a classification based on size and aspect ratio of connected components, but this clearly can be improved. That step also identifies explicit column separators, such as long, thin vertical lines between columns that occur in some layouts.

To generate the reflowable document image, the individual text lines need to be subdivided further into "words". It is not absolutely critical that these subdivisions correspond exactly to words; failure to break two words apart will just lead to a slightly more ragged appearance of the reflowable document, while breaking within a word leads is not visible unless, by chance, the reflowable image is displayed at a width that makes breaking at that point convenient.

We refer to the collection of word bounding boxes, text line segments, and reading order annotations as the "reflow annotations" for the image. When the reflow annotations are bundled with the original document image, the resulting bundle contains all the information necessary for reflowing the image for display in different window sizes and on different devices.

These bundles can actually be viewed directly using a suitable custom viewer. Such a representation is nice because it requires no modification to the original document image file. This may be important in some applications. It


Mozilla
File Edit View Go Bookmarks Tools W
Mozilla
File Edit View Go Bookmarks Tools

Widely distributed in North America. Little known in California. Locs.—Oro Fino, *Butler* 137; Russian River, s. Mendocino Co., *Heller* 5827 (det. C. V. Piper); Chinatown firth, Santa Ana River, *F. M. Reed* (acc. Agnes Chase). Probably over looked elsewhere in California.

Refs.—Scirpus validus Vahl, Enum. Pl. 2:268 (1806), type from the West Indies. *S. lacustris* of Am. authors.

9. **S. americanus** Pers. Three Square. (Fig. 20.) Stems ¾ to 2 feet high, very slender, triangular, some what leafy; leaves short (the blade 1 to 3 inches long); involucral bract solitary, pungent, 1 to 4 inches long; spikelets 1 to 6, oblong-ovate, 3 to 7 lines long, borne in a single crowded sessile cluster; scales dark-brown, usually conspicuously tipped with a stout pale-colored awn about a line long; achene flat on one face, convex on the other and somewhat obscurely keeled; bristles 2 to 6, very unequal, the longer about as long as the achene.

Fig. 20. Scirpus americanus Pers. a, cluster of spikelets, × 1; b, scale, × 5; c, achene and bristles, × 5.

Marshy, often brackish, places, occasional throughout California. North America, Chile.

Locs.—Panamint Cañon, *Hall & Chandler* 7041; Owens Lake, *Jepson* 5115; Mt. Pinos, *Hall* 6627; Eureka, *Tracy* 1765; Castle Rock, Sacramento River, *Goldsmith* 7; Honey Lake Valley, *Davy*

Piper); Chinatown firth, Santa Ana River, *F. M. Reed* (acc. Agnes Chase). Probably over looked elsewhere in California.

Refs.—Scirpus validus Vahl, Enum. Pl. 2:268 (1806), type from the West Indies. *S. lacustris* of Am. authors.

9. **S. americanus** Pers. Three Square. (Fig. 20.) Stems ¾ to 2 feet high, very

Figure 3. Multiple renditions of the same document in Mozilla on a desktop machine. Mozilla can easily handle reflowing this content in real-time on this 1GHz PC running Linux. Different text sizes are available.

also shows us that reflowable document images have a fairly modest overhead compared with the original document image, since the only additional information required is the reflow annotations. These amount to a few kilobytes of data per page.

Bundles can be converted into a variety of other formats (Figure 2). The PDF format recently has been enhanced by Adobe to permit the use of reflow information. A direct conversion of bundles into reflowable PDF may be possible. Similarly, it may be possible to convert bundles into structured vector graphics, together with JavaScript code to handle the reflowing. However, since both of those technologies are not widely deployed yet, they will not be covered here.

3 HTML-Based Representations

HTML as an Intermediate Format HTML turns out to be a fairly natural representation for reflowable document images. A reflowed document image is shown at two sizes in a desktop browser in Figure 3. And the same document can be rendered easily on a PDA (Figure 4)

Reflowable document images represented as HTML consist of a long sequence of image references. These usually alternate between references to word images and references to a single image representing pixel-accurate interword spacing.

Optional Spacing One problem that limits the fidelity of reflowable document images represented as HTML is the limited control that HTML gives over spacing. In particular, a high quality rendition requires the use of inter-word spacing that disappears when the text line is broken at that position. If that space is retained at the end of a text line, it is usually not very noticeable, but if it flows into the next line, it appears as an indentation (this latter case can be avoided by using the non-standard `NOBR` tag or equivalents).

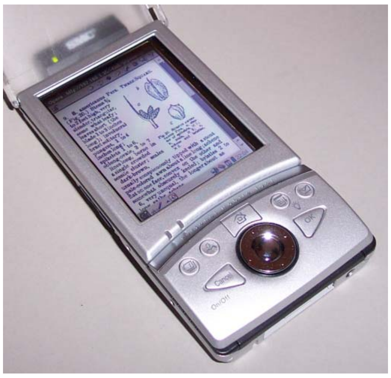


Figure 4. Display of reflowable images on a Sharp Zaurus at 240×320 pixels.

While HTML does perform this kind of handling of space for space characters, those characters are not of predictable width and cannot be used for the inter-word spacing in reflowable document images.

Hyphenation Related to optional spacing is the issue of optional hyphenation. In Western languages, documents that have been rendered onto a page will contain hyphens to indicate words that have been broken across lines. These hyphenations are easy to detect even without full OCR, but it is impossible to determine based on appearance alone whether the hyphen is an essential part of the word (as in “re-examine”) or whether it is only present because the word has been broken across lines (some Western languages, however, use separate symbols for the two cases). But even if it were possible to detect these cases, unlike other type setting languages, it is not possible to indicate optional hyphens in current versions of HTML. Fortunately, these cases are rare and do not appear to affect readability.

Ultimately, both hyphenation and word breaks might be identified by attempting OCR and outputting word breaks and optional hyphens where the OCR reliably identified words, with graceful degradation to the current appearance-based mechanisms. However, until the facilities to actually perform high-quality rendering of these subtle features in HTML, there is little point to computing them, at least for web-based display.

Network Performance One of the most serious limitations of HTML based representations of reflowable document images is the impact on network performance. The running example used in this paper is composed of about 500 word images per page. These currently result in 500 separate requests from the web server. Features like HTTP keep-alive connections, pipelining, compression, and parallel requests help make loading reflowable document images even over remote web connections feasible, but performance improvements are desirable.

Ideal would be if a web page archive became standard that allowed the HTML and all associated images to be downloaded in a single transaction. Such a format actually already exists and is used, for example, in HTML-based help systems and for web page archiving. However, it is not (yet?) widely supported as an Internet file format.

Another approach to keeping the amount of data down would be to download the original source image (e.g., in PNG format), together with an HTML file that references sub-rectangles of that image for rendering. This would reduce the number of transactions to two and require little more overhead than the display of the original document image.

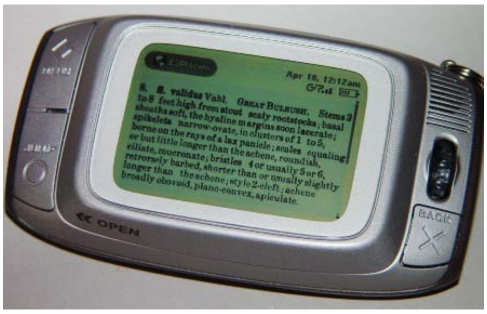
In principle, a combination of style sheets and JavaScript is capable of achieving this. However, the currently available implementations of dynamic HTML in major web browsers (Microsoft Internet Explorer and Mozilla) do not implement this feature consistently and reliably[1].

Web sites increasingly use large numbers of small image components, and this places a strain on web servers. Therefore, it is likely that these issues will be addressed soon, making HTML a fairly efficient and ubiquitous representation of reflowable document images.

Java and Flash Java and Flash are two systems that could be programmed to interpret and render bundles of reflow annotations and document images, getting around the limitations of HTML for these purposes. Furthermore, Java and Flash are widely available on desktop machines. However, their integration with the web browser is still too limited to be able to achieve a natural user experience. For example, embedded Java or Flash applets do not easily resize along with the containing page.

HTML Chunking and Proxying Some devices (e.g., the Danger Hiptop phone, Figure 5) claim to be able to handle general HTML, but their display engines are overtaxed by pages containing large numbers of images.

A solution to both this problem and the problem of large numbers of small HTTP transactions is to perform some of the reflow computations on the server. This is feasible because screen sizes, in particular for handheld devices, generally only occur in a fixed number of widths. For example, for the Danger Hiptop, we can precompute, on the server, a set of reflowed document images for its specific display width (240 pixels). Then, instead of sending individual word images, we can send blocks of multiple text lines, reflowed to the device width, as images.



Apr 18, 12:12am

6. S. validus Vahl. Great Bulrush. Stems 3 to 8 feet high from stout scaly rootstocks; basal sheaths soft, the hyaline margins soon lacerate; spikelets narrow-ovate, in clusters of 1 to 5, borne on the rays of a lax panicle; scales equaling or but little longer than the achene, roundish, ciliate, mucronate; bristles 4 or usually 5 or 6, retrorsely barbed, shorter than or usually slightly longer than the achene; style 2-cleft; achene broadly obovoid, plano-convex, apiculate.

Figure 5. Display of reflowable images on the Danger Hiptop, a cell phone with a screen resolution of 240×160.

Alternatively, this kind of chunking can be performed dynamically by a proxy server that transforms the bundles (original page images together with reflow annotations) into correctly sized image chunks on the fly.

4 Discussion

This paper has described on-going work in developing systems for creating reflowable document images. These are formats that are intermediate between structured text and pure image-based formats. They retain the appearance of the original text but can contain content that cannot be captured reliably by OCR or even represented in non-image formats for many browsers. The layout analysis techniques developed as part of this work are also applicable to the creation of other electronic book formats (either image based or OCR-based). But conversion to HTML in particular promises to make scanned documents available universally in a format that is non-proprietary, easy to implement, and convenient for end users. As discussed above, HTML still has some minor limitations for this application, but these will likely get addressed as HTML and browser implementations mature further.

References

[1] T. M. Breuel. Two algorithms for geometric layout analysis. In *Proceedings of the Workshop on Document Analysis Systems, Princeton, NJ, USA*, 2002.

[2] T. M. Breuel, W. C. Janssen, K. Popat, and H. S. Baird. Paper-to-pda. In *Proceedings of the International Conference on Pattern Recognition (ICPR'02), Quebec City, Quebec, Canada*, 2002.

[3] T. M. Breuel, W. C. Janssen, K. Popat, and H. S. Baird. *Web Document Analysis: Challenges and Opportunities, A. Antonacopoulos and Jianying Hu, editors,* chapter Reflowable Document Images. 2002.

[1] However, absolute positioning works well in current implementations, so it is easy to present a reflowable document image represented as a large collection of separate word images pixel-accurately in its original form by positioning the individual words at their exact locations.